

# Cache Performance Analysis for Block Size Selection in the ATLAS Library

Neungsoo Park, Bo Hong,  
and Viktor K. Prasanna  
University of Southern California

June 2002  
<http://ceng.usc.edu/~prasanna>

# Outline

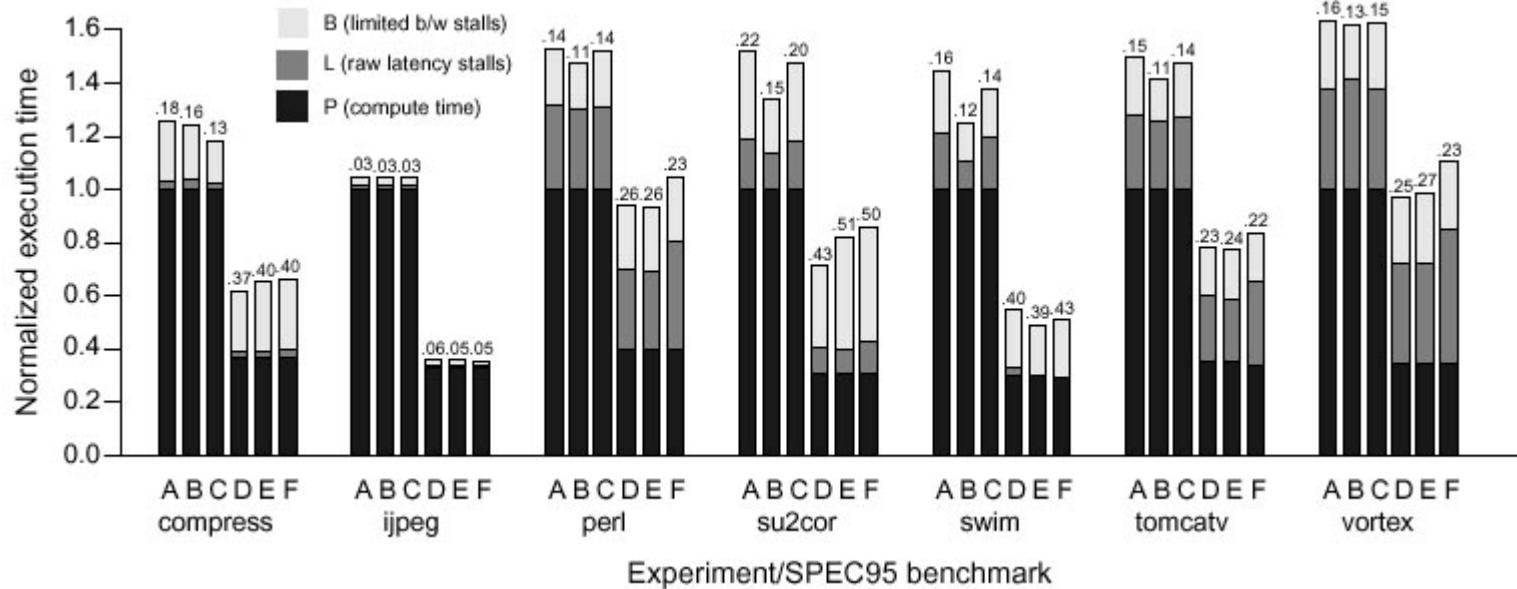
---

- Motivation
- TLB and Cache Performance of Block Data Layout
- Block Size Selection in ATLAS
- Optimizing FFT Performance
- Concluding Remarks

# Motivation

Table 2. Processor simulation parameters.

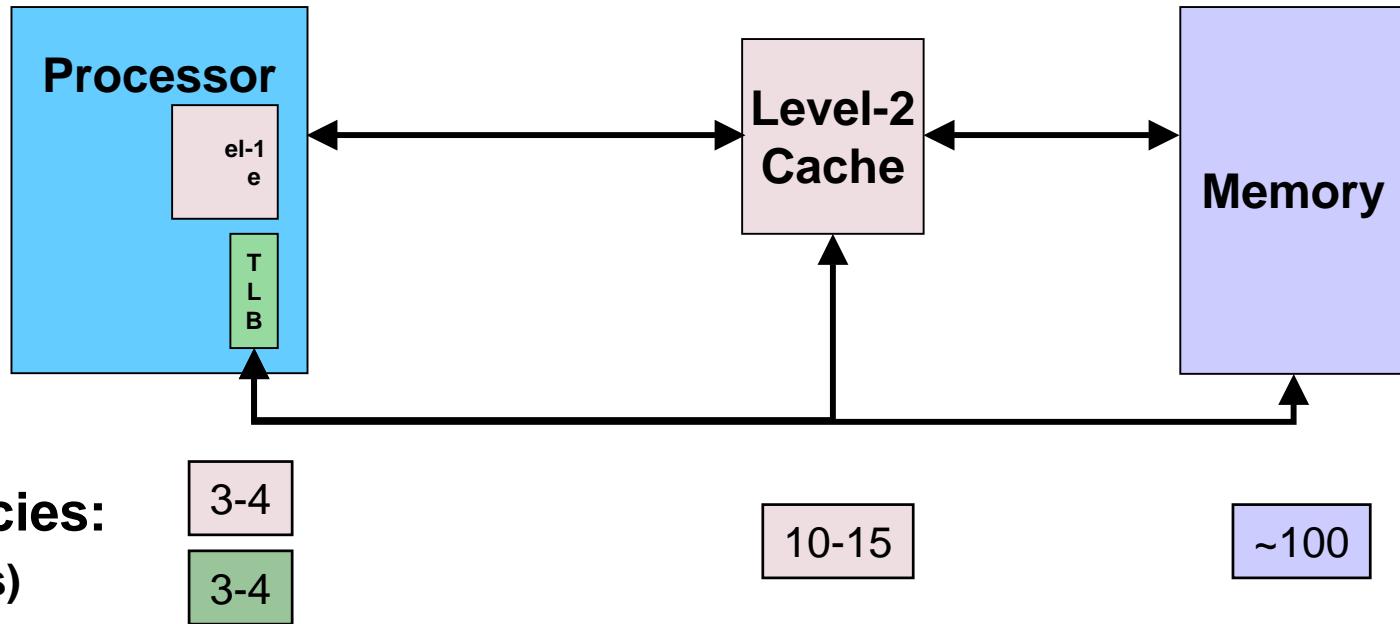
| Parameter         | Experiment     |        |   |             |                    |   |
|-------------------|----------------|--------|---|-------------|--------------------|---|
|                   | A              | B      | C | D           | E                  | F |
| Processor         | In-order issue |        |   |             | Out-of-order issue |   |
| Clock speed       | 500 MHz        |        |   |             | 1 GHz              |   |
| RFU slots         | 128            |        |   |             | 256                |   |
| L5 Entries        | 64             |        |   |             | 128                |   |
| Branch predictor  | 16K            |        |   |             | 32K                |   |
| Caches            | Blocking       |        |   | Lockup-free |                    |   |
| L1/L2 block sizes | 32/51          | 64/128 |   |             | 32/51              |   |
| HW prefetch       | No             |        |   | Yes         |                    |   |



- Typical Sustained Performance  $\approx 10\%$

\* Doug Burger, et. al. "Limited bandwidth to Affect Processor Design", *IEEE Micro*, Nov. 1997.

# The Problem



- Processor speed relative to memory speed (processor-memory gap) growing rapidly
- Data and/or page table entry not present in level-1 cache and/or TLB causes severe memory stalls in processor
- Data set size for common applications growing

# Memory Access Costs

---

- DEC Alpha 21164
  - on-chip cache: ~ 10 nsec latency
  - off-chip cache: 26 nsec latency
  - main memory: 253 nsec latency
- UltraSPARC III
  - on-chip level-1 cache: 3-4 cycles
  - on-chip level-2 cache: ~15 cycles
  - main memory: ~100 cycles

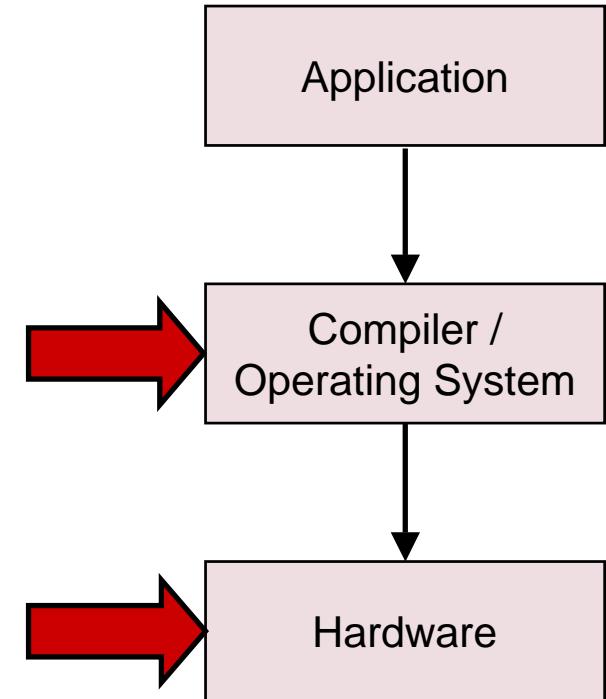
# State of the Art

- Software (compiler / operating system)

- Code reordering to improve locality and hide memory stalls
- Prefetching to hide memory stalls
- Tiling to improve memory locality
- **Compiler cannot always unwrap data dependencies**  $\Rightarrow$  tiling not possible
- Pointer chasing difficult to prefetch at run time or optimize at compile time

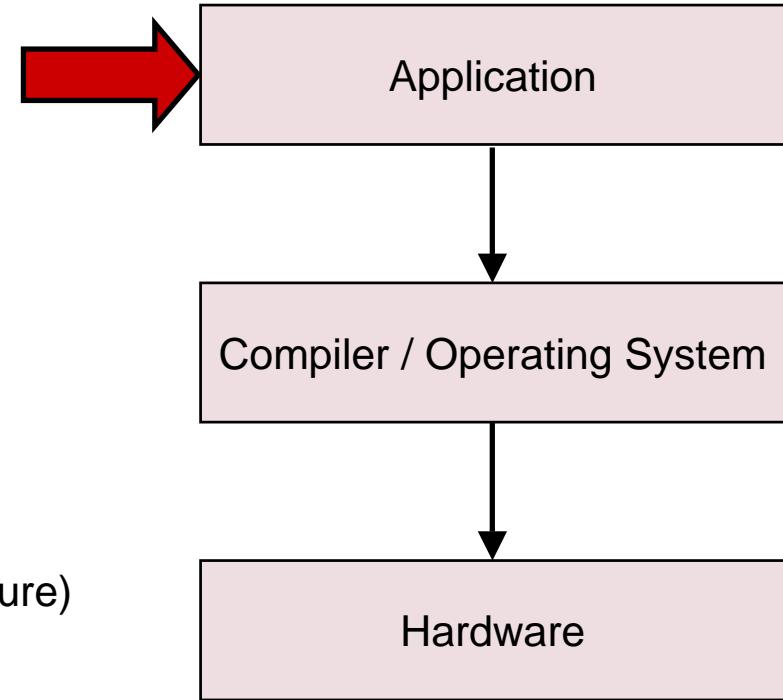
- Hardware

- Increase memory bandwidth to shorten latencies
- Higher associativity to avoid cache conflicts
- **Processor outpacing memory in speed**
- **Applications requiring larger data sets**  
 $\Rightarrow$  require larger cache to achieve equal cache performance



# Cache Conscious Algorithm Design: Our Focus

- No control over compiler / operating system
- No control over hardware
- Novel Algorithmic Techniques
  - Examine complete algorithm to perform data access reordering
    - Reduce cache pollution
    - Increase data reuse
  - Data layouts tuned to the access pattern (Single and/or multiple data structure)
    - Reduce cache pollution
    - Increase data reuse
    - Reduce cache conflicts
    - Reduce TLB thrashing

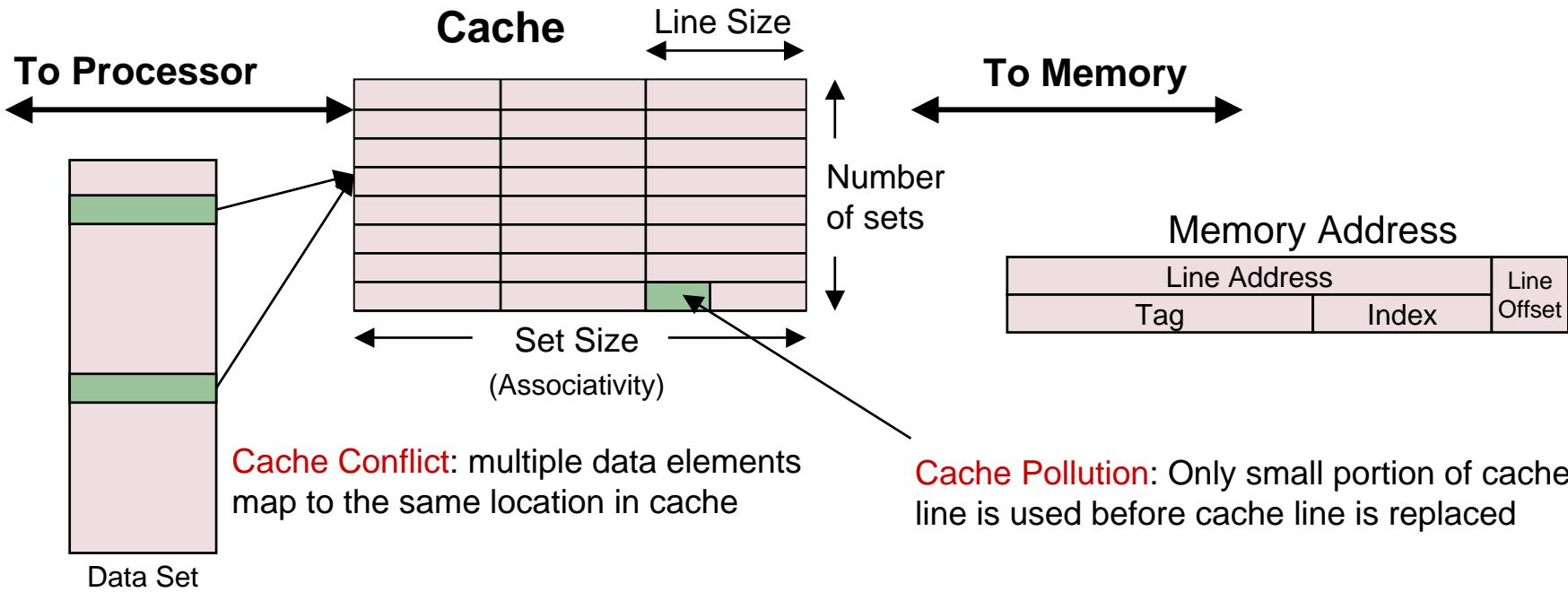


# Outline

---

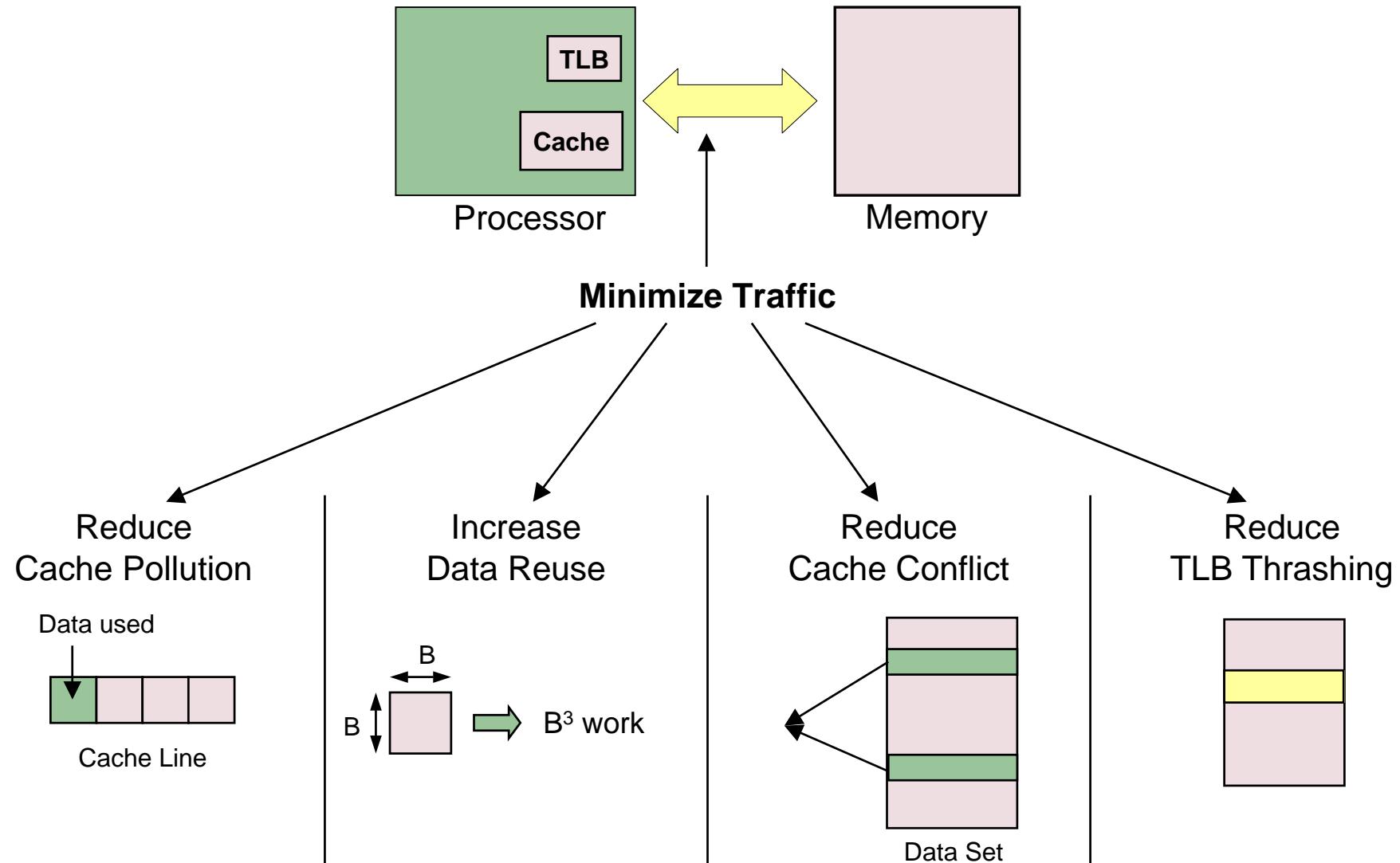
- Motivation
- TLB and Cache Performance of Block Data Layout
- Block Size Selection in ATLAS
- Optimizing FFT Performance
- Concluding Remarks

# Cache Memory Concepts



- Definitions
  - **Line Size:** Amount of data transferred from memory after a cache miss
  - **Set Size / Associativity:** Available places in the cache for a cache line to be placed
  - **Number of Sets:** cache size / (line size \* set size)
  - **Replacement policy:** policy used by memory system to decide which cache line in a set to replace during a cache miss
  - **Line Index:** Used to select set in which to place cache line
  - **Line Tag:** Used to detect presence of correct address in cache
  - **Line Offset:** Indicates where in the cache line a memory address is located

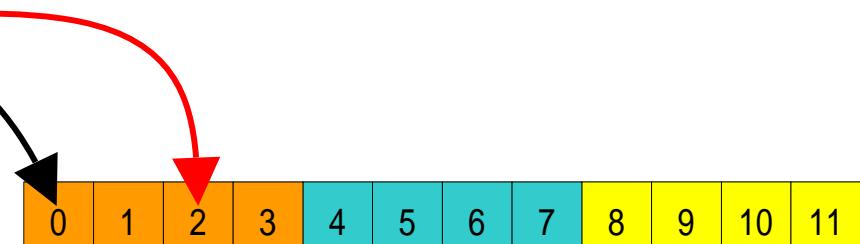
# Cache Conscious Algorithm Design: Techniques



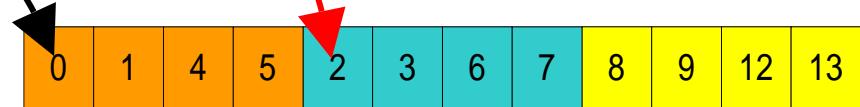
# Block Data Layout (BDL)

## Row-major Layout

|    |    |    |    |
|----|----|----|----|
| 0  | 1  | 2  | 3  |
| 4  | 5  | 6  | 7  |
| 8  | 9  | 10 | 11 |
| 12 | 13 | 14 | 15 |



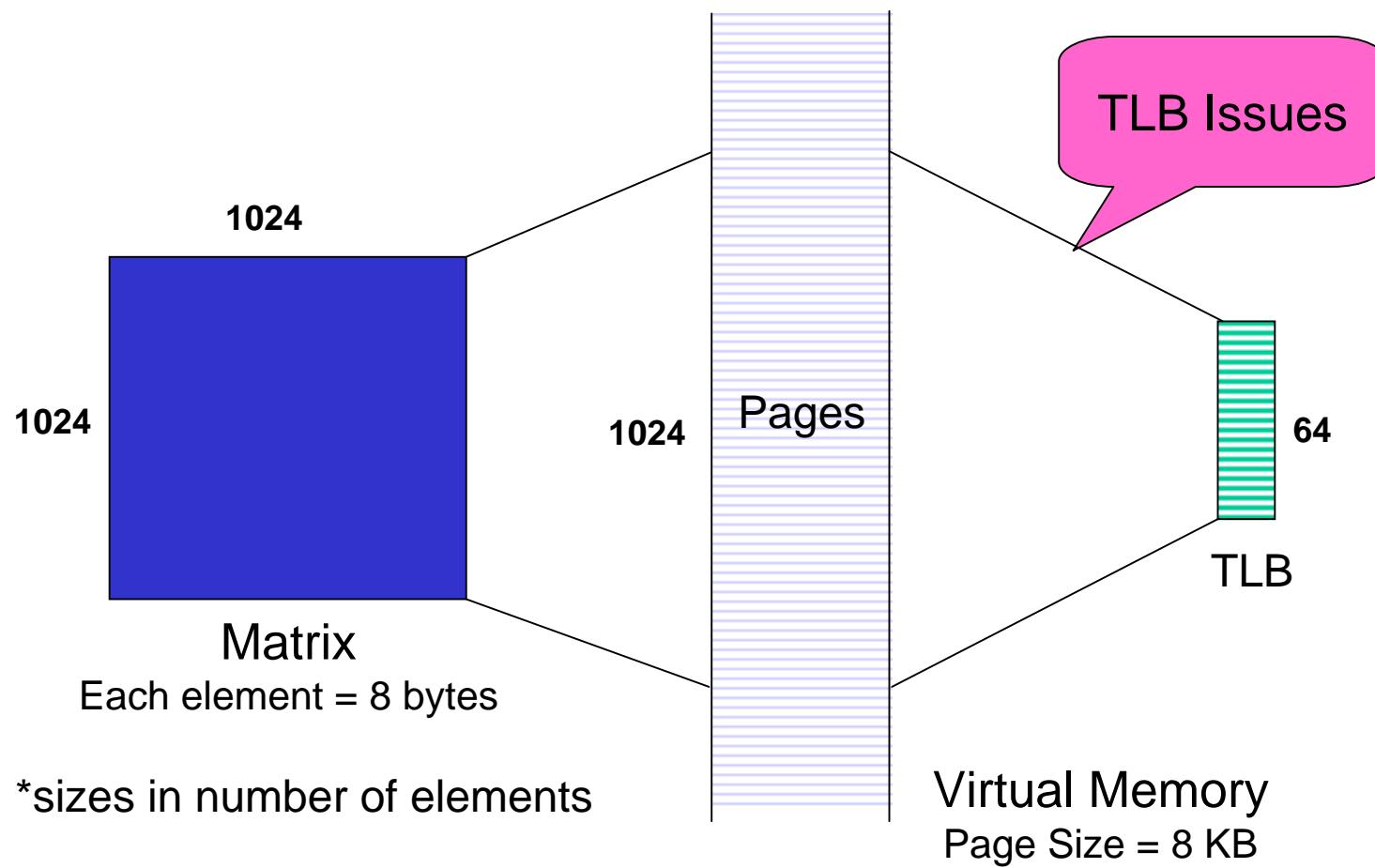
|    |    |    |    |
|----|----|----|----|
| 0  | 1  | 2  | 3  |
| 4  | 5  | 6  | 7  |
| 8  | 9  | 10 | 11 |
| 12 | 13 | 14 | 15 |



Mapping onto linear memory space

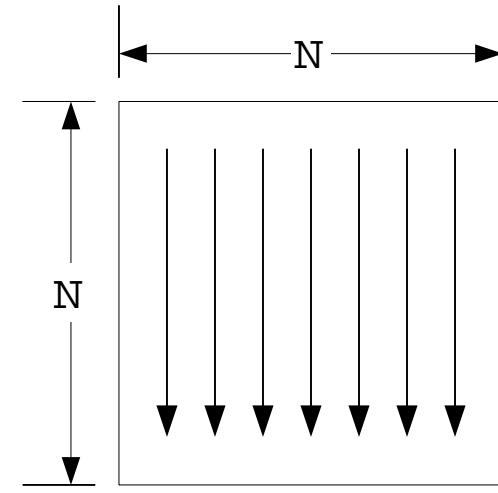
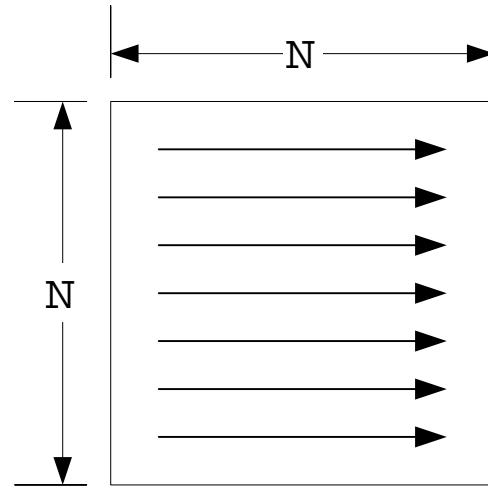
## Block Data Layout

# Mapping Data in the Memory Hierarchy



# Generic Access Pattern

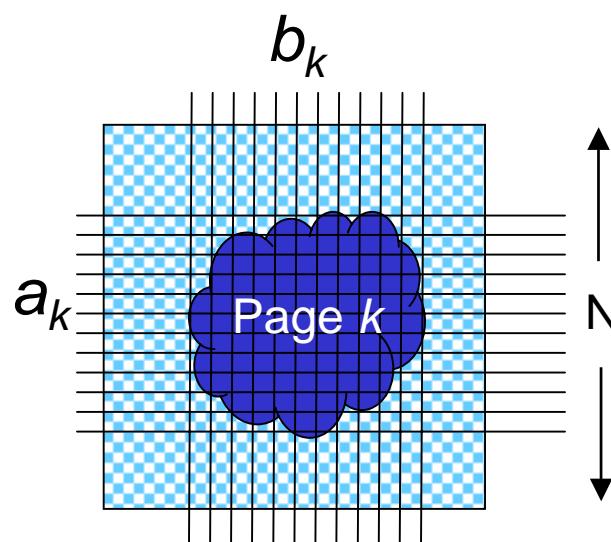
- $N \times N$  matrix



- $N$  row accesses and then  $N$  column accesses
- $P_v$  elements can be stored per page
- $N >> \text{TLB Size } (S_{\text{tlb}})$

# A Lower Bound on TLB Misses (1)

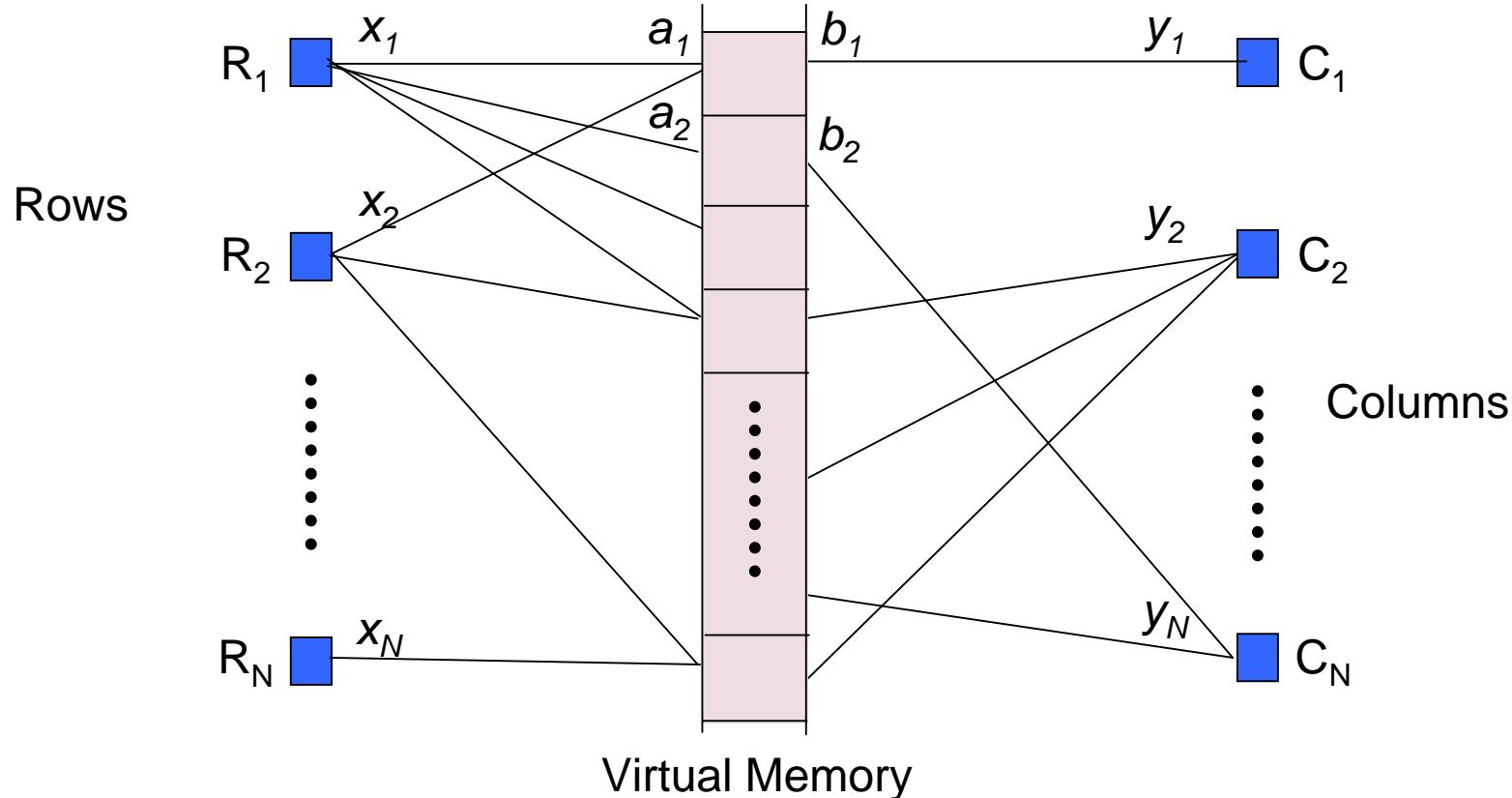
**Theorem:** For accessing an array along all the rows and then along all the columns, the asymptotic minimum number of TLB misses is given by  $2N^2/\text{sqrt}(P_v)$ .



- $A_k = \{i \mid \text{at least one element of row } i \text{ is in page } k\}$
- $B_k = \{j \mid \text{at least one element of column } j \text{ is in page } k\}$
- $a_k = |A_k|$  and  $b_k = |B_k|$
- $a_k \times b_k \geq P_v$
- $a_k + b_k \geq 2\text{sqrt}(a_k b_k) \geq 2\text{sqrt}(P_v)$

# A Lower Bound on TLB Misses (2)

- Arbitrary mapping of array to pages
  - $x_i$  = # of pages where row  $i$  is mapped
  - $y_j$  = # of pages where row  $j$  is mapped



# A Lower Bound on TLB Misses (3)

- The Number of TLB misses for all row and then all column accesses

$$TLB_{miss} \geq \sum_{i=1}^N (x_i - O(S_{tlb})) + \sum_{j=1}^N (y_j - O(S_{tlb}))$$

$$TLB_{miss} \geq \sum_{k=1}^{N^2/P_v} (a_k + b_k) - 2N \bullet O(S_{tlb}) \geq 2 \frac{N^2}{\sqrt{P_v}} - 2N \bullet O(S_{tlb})$$

$O(S_{tlb})$ : # of TLB entries that are reused

Since  $N \gg S_{tlb}$ ,

$$\therefore TLB_{miss} \geq 2 \frac{N^2}{\sqrt{P_v}}$$

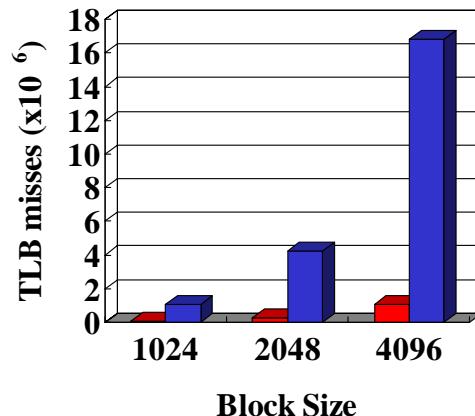
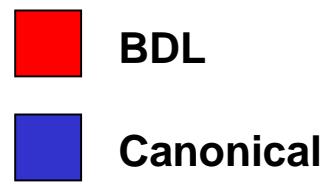
# TLB Performance Using BDL (1)

**Theorem:** *For accessing an array along all the rows and then along all the columns, block data layout with block size  $\sqrt{P_v} \times \sqrt{P_v}$  minimizes the number of TLB misses.*

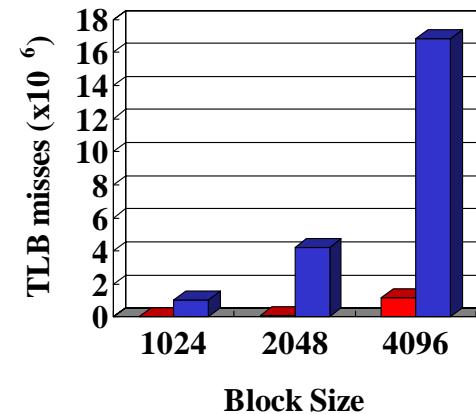
# TLB Performance Using BDL (2)

- Simulation

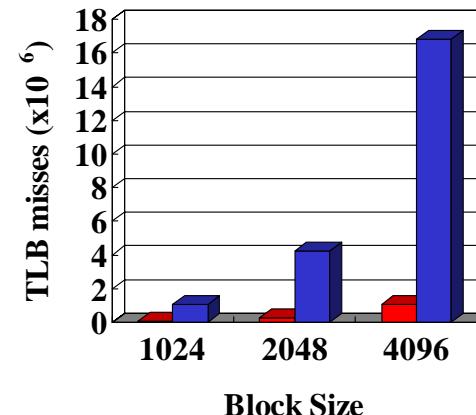
- Page size: 8 KB
- TLB
  - fully set-associative
  - 64 entries
- SimpleScalar simulator
- Block size: 32 x 32



(b) Arbitrary permutation of row and column accesses



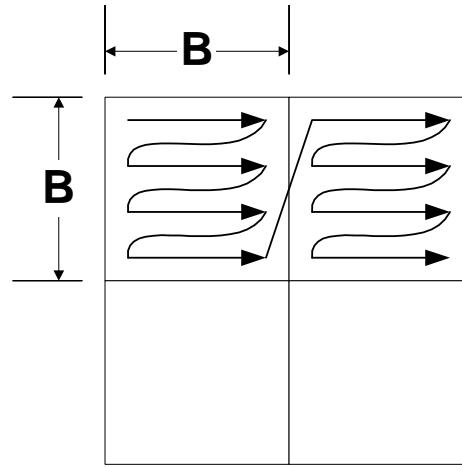
(a) Along all rows and then all columns



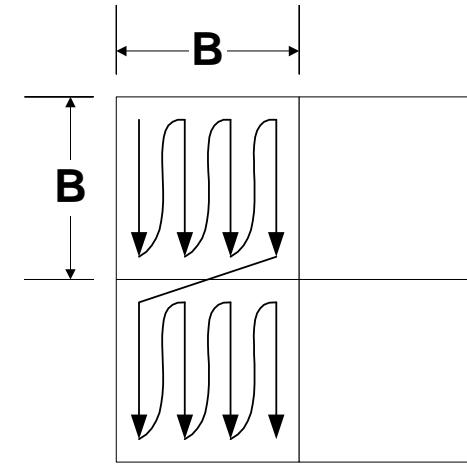
(c) Arbitrary permutation of all rows followed by arbitrary permutation of all columns accesses

# Tiling + BDL

- Tiled Access



(a) Tiled row access

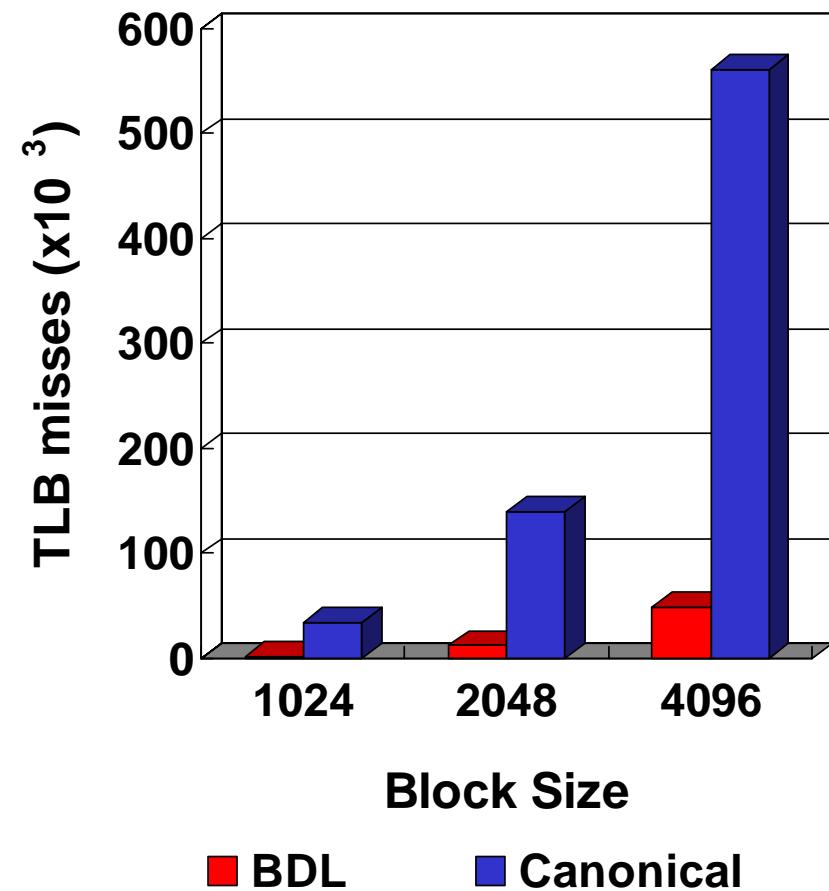


(b) Tiled column access

- Block size:  $B \times B$

# TLB Misses Using Tiling + BDL

- TLB misses for all tiled row accesses followed by all tiled column accesses
- Simulation
  - Page size: 8 KB
  - TLB
    - fully set-associative
    - 64 entries
  - SimpleScalar simulator
  - Block size: 32 x 32



# Tiled Matrix Multiplication (TMM) (1)

- 6-loop Tiled Matrix Multiplication

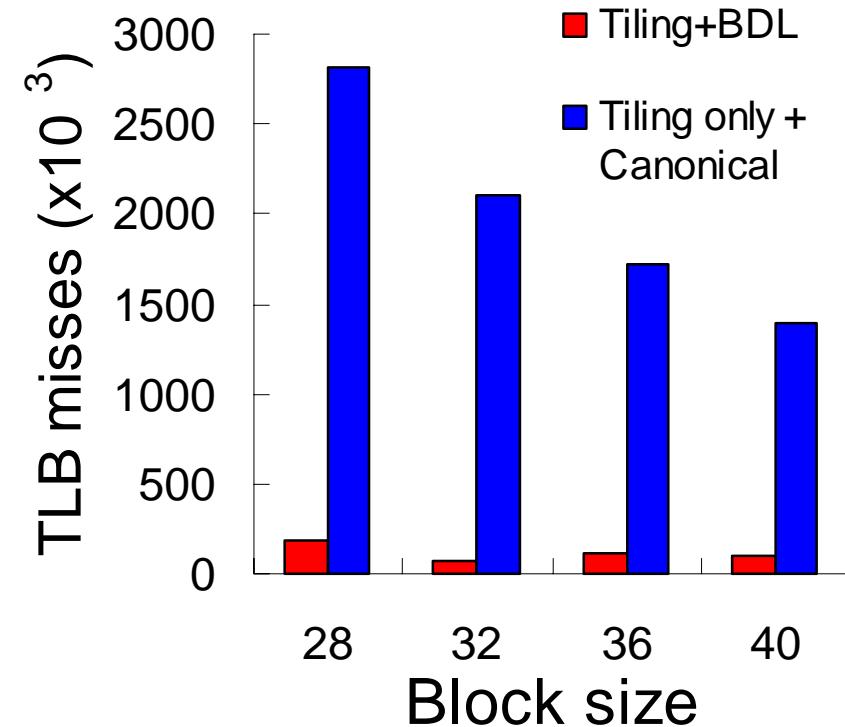
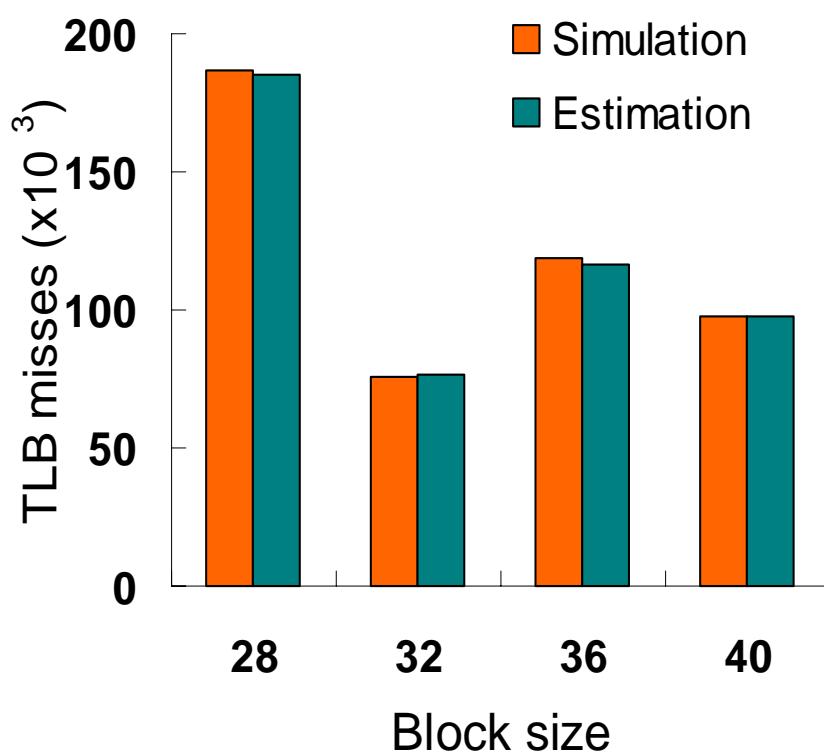
```
for jj=0 to N by B
    for kk=0 to N by B
        for ii=0 to N by B
            for i=ii to min(ii+B-1,N)
                for k=kk to min(kk+B-1,N)
                    r = A(i,k)
                    for j=jj to min(jj+B-1,N)
                        C(i,j) += r*B(k,j)
```

- Number of TLB misses:

$$TM = 2N^3 \left( \frac{1}{BP_v} + \frac{1}{B^3} \right) + N^2 \left( \frac{1}{P_v} + \frac{1}{B^2} \right)$$

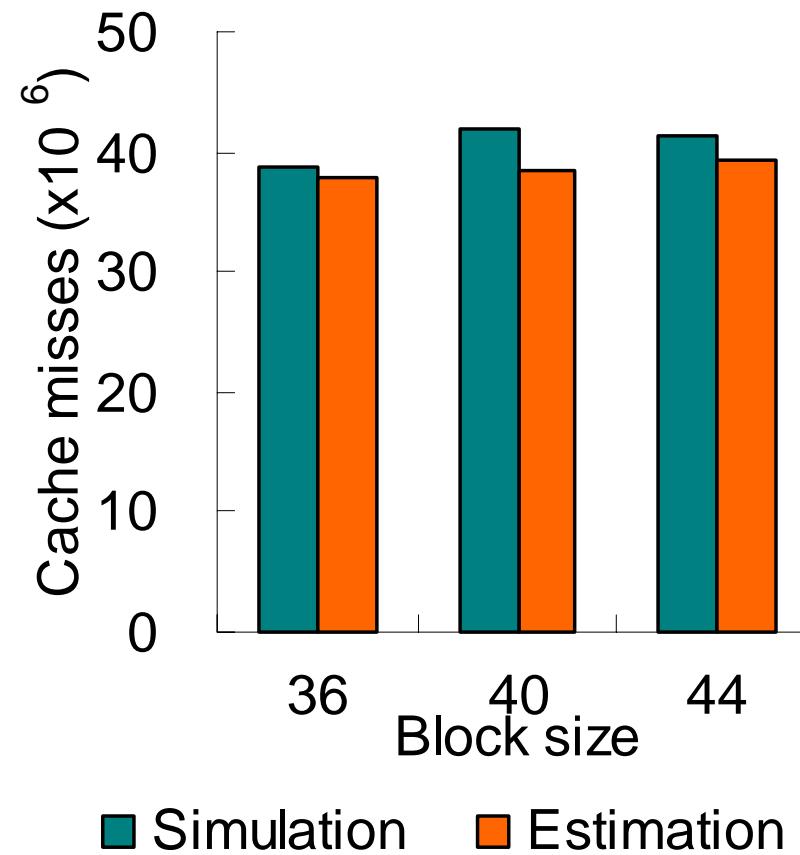
# Tiled Matrix Multiplication (TMM) (2)

- TLB Misses of Tiling + BDL



# Tiled Matrix Multiplication (TMM) (3)

- Cache miss equations
  - Similar to that using tiling+copying
- Cache simulation
  - SimpleScalar simulator
  - Problem size
    - $1024 \times 1024$
  - Cache
    - Size: 16 KB
    - Line size: 32 bytes
    - Direct-mapped



# Outline

---

- Motivation
- TLB and Cache Performance of Block Data Layout
- **Block Size Selection in ATLAS**
- Optimizing FFT Performance
- Concluding Remarks

# Block Size Selection (1)

- Total Miss Cost

$$MC = TM \bullet M_{tlb} + \sum_{i=1}^j CM_i \bullet H_{i+1}$$

# of TLB misses      TLB miss cost      # of  $L_i$  cache misses      Hit cost of  $L_{i+1}$  cache

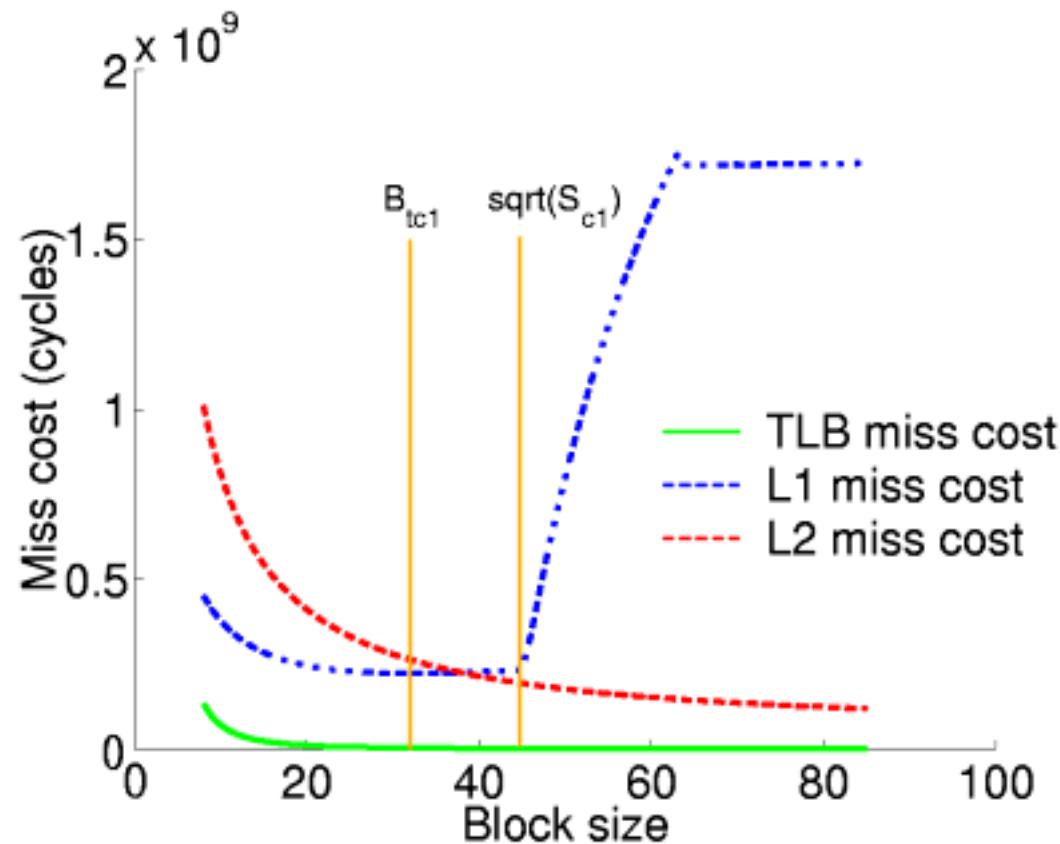
- Range of optimal block size for TMM

$$B_{tc1} \leq B_{opt} < \sqrt{S_{c1}}$$

- $S_{c1}$  :  $L_1$  cache size
- $B_{tc1}$ : the block size to minimize the miss cost caused by TLB and L1 cache misses

# Block Size Selection (2)

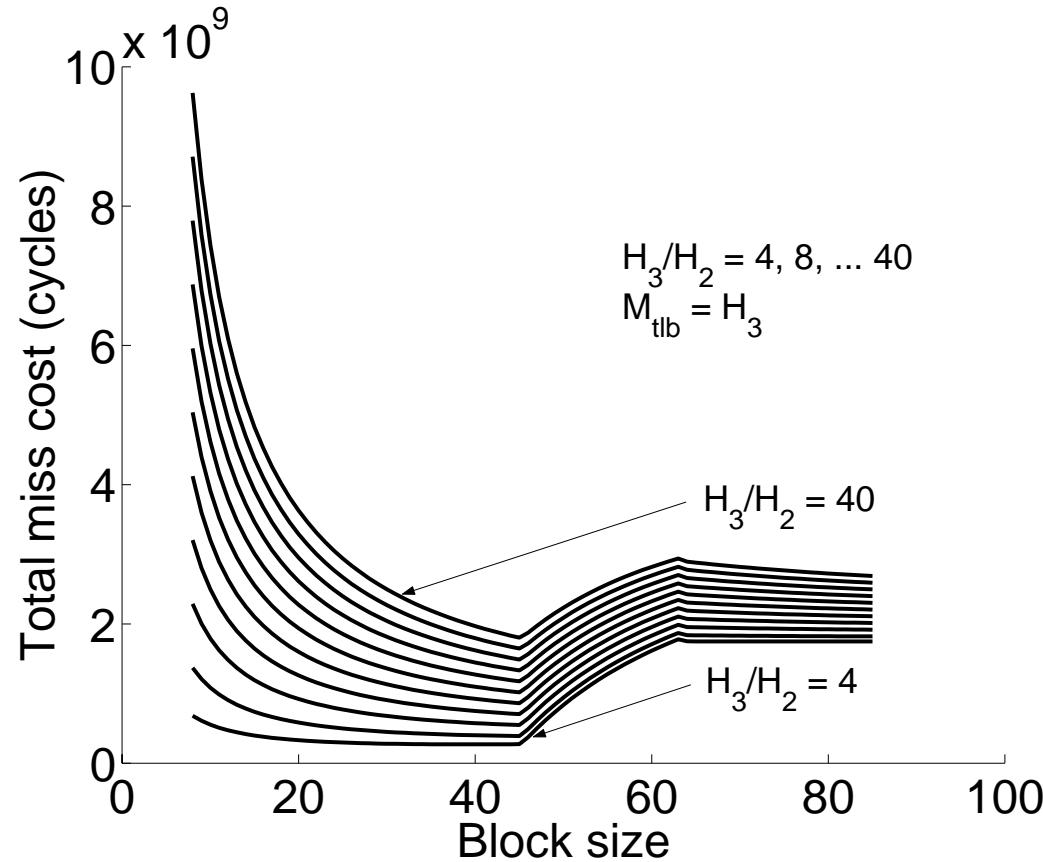
- The total miss cost for TMM



(a) Miss cost of TLB, L1, and L2 cache

# Block Size Selection (3)

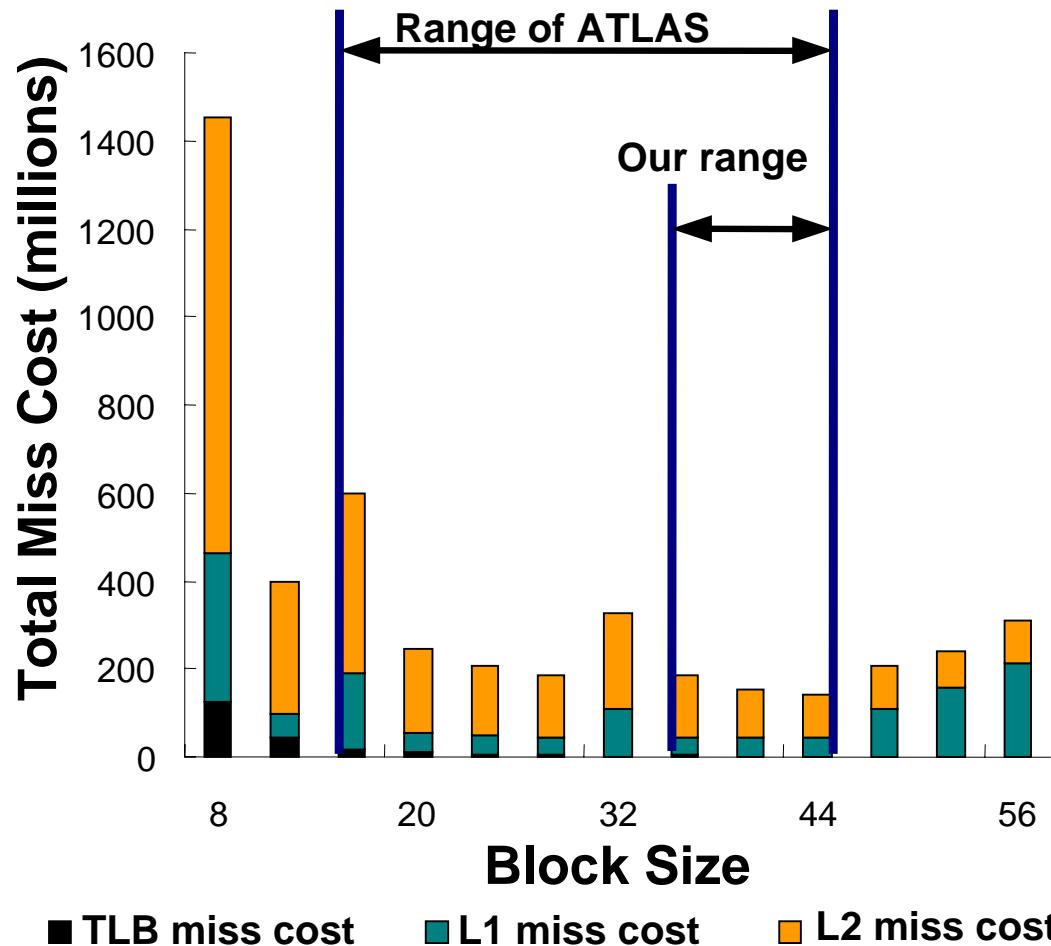
- The total miss cost for TMM



(b) Miss cost of TLB, L1, and L2 cache

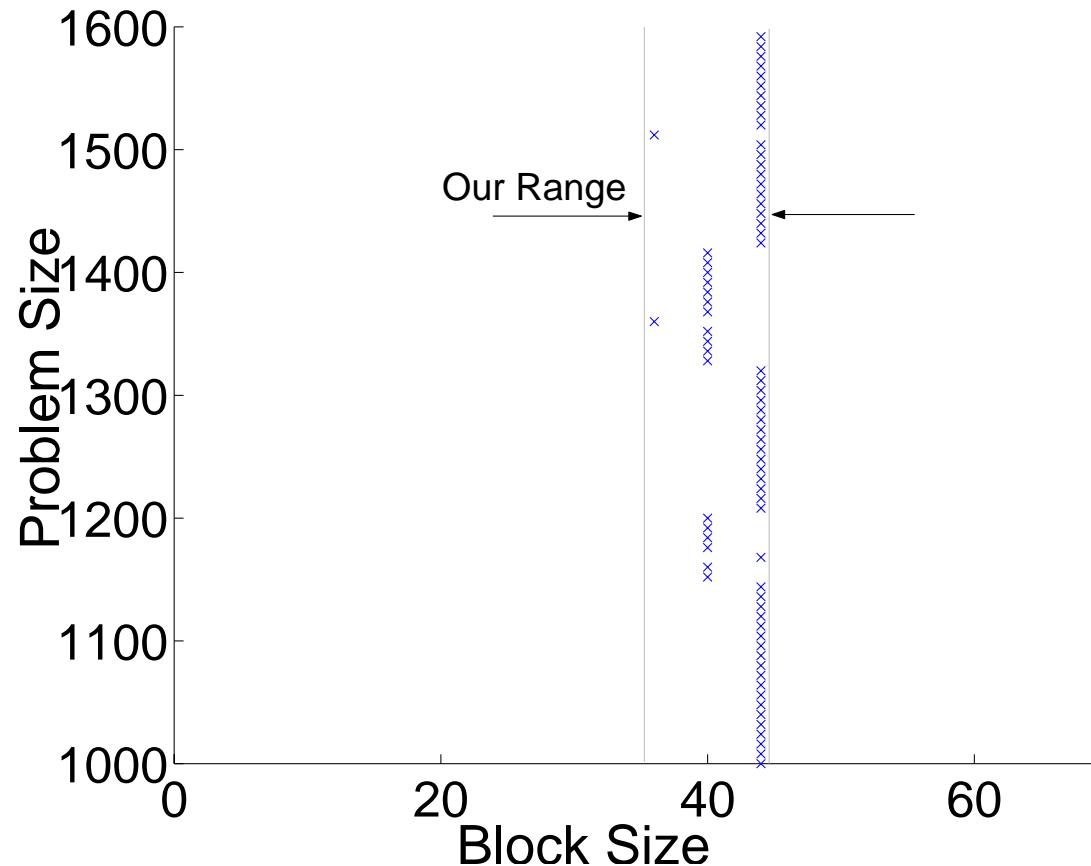
# Optimal Block Size for TMM (1)

- Simulation using UltraSPARC II parameters



# Optimal Block Size for TMM (1)

- Experiments on UltraSPARC II



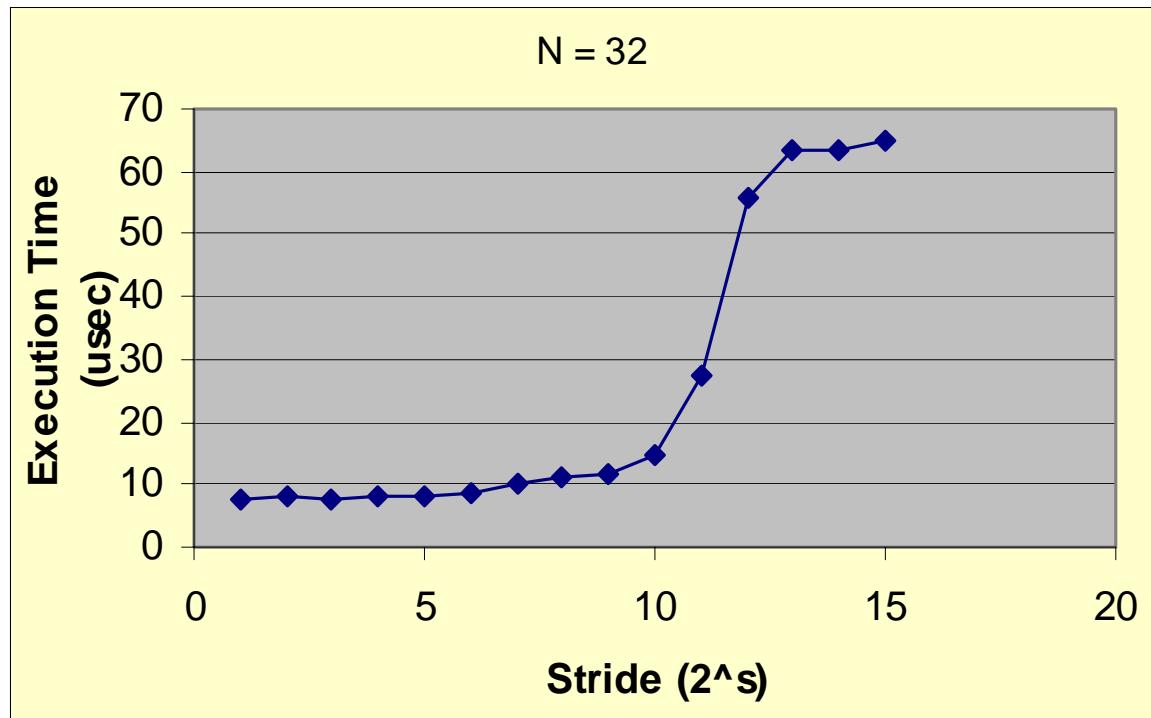
# Outline

---

- Motivation
- TLB and Cache Performance of Block Data Layout
- Block Size Selection in ATLAS
- Optimizing FFT Performance
- Concluding Remarks

# Example: FFT with Stride Input Data

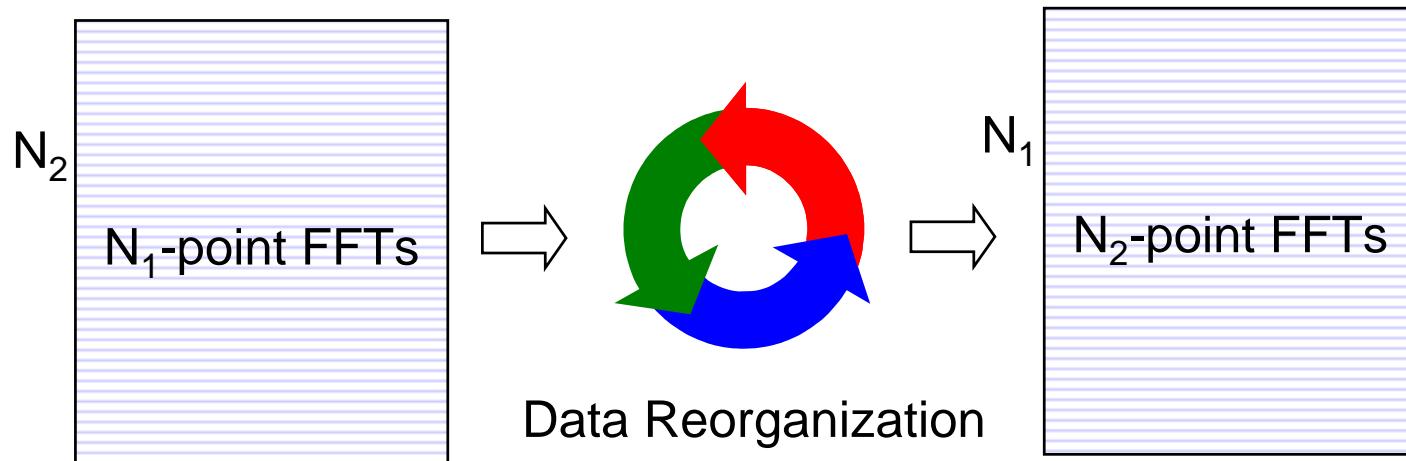
Time to compute 32-point DFT



Sun Ultra 1: 167MHz, L2 Cache = 512 KB = 32 K points (gcc -O3)

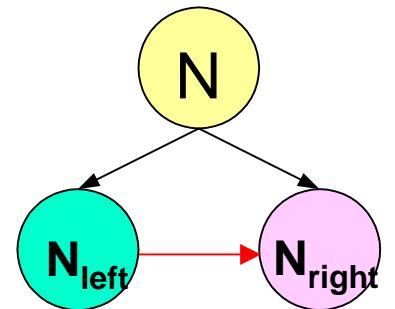
# Our Approach

- Reorganize input data layout from non-unit stride to unit stride
- Perform data reorganization during computation  $\Rightarrow$  **Dynamic Data Layout**



# Optimal Factorization Tree

## Recursive Factorization



$$\begin{aligned} N &= 2^n \\ &= N_{\text{left}} \times N_{\text{right}} \end{aligned}$$

Data Reorganization ?

- Huge search space
  - $O(4^n)$  possible factorization trees
- **Dynamic programming** determines the optimal factorization tree
  - Consider the **size** of the transform and the **stride** of data access
  - $k$  different layouts for each node
  - Search complexity:  $O(kn^2)$

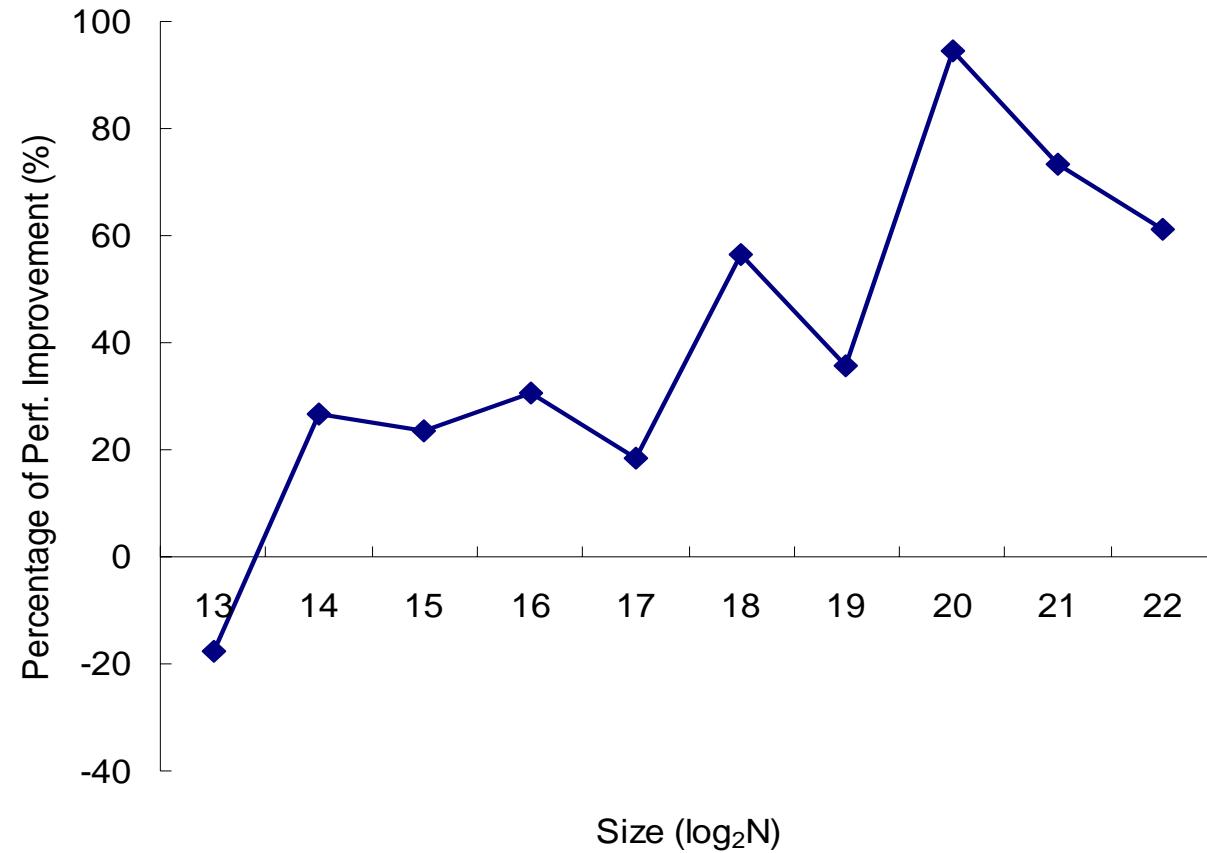
# Sample Experimental Results: FFT with Dynamic Data Layouts

- Performance Comparison with MIT's FFTW  
**(Fastest Fourier Transform in the West)**

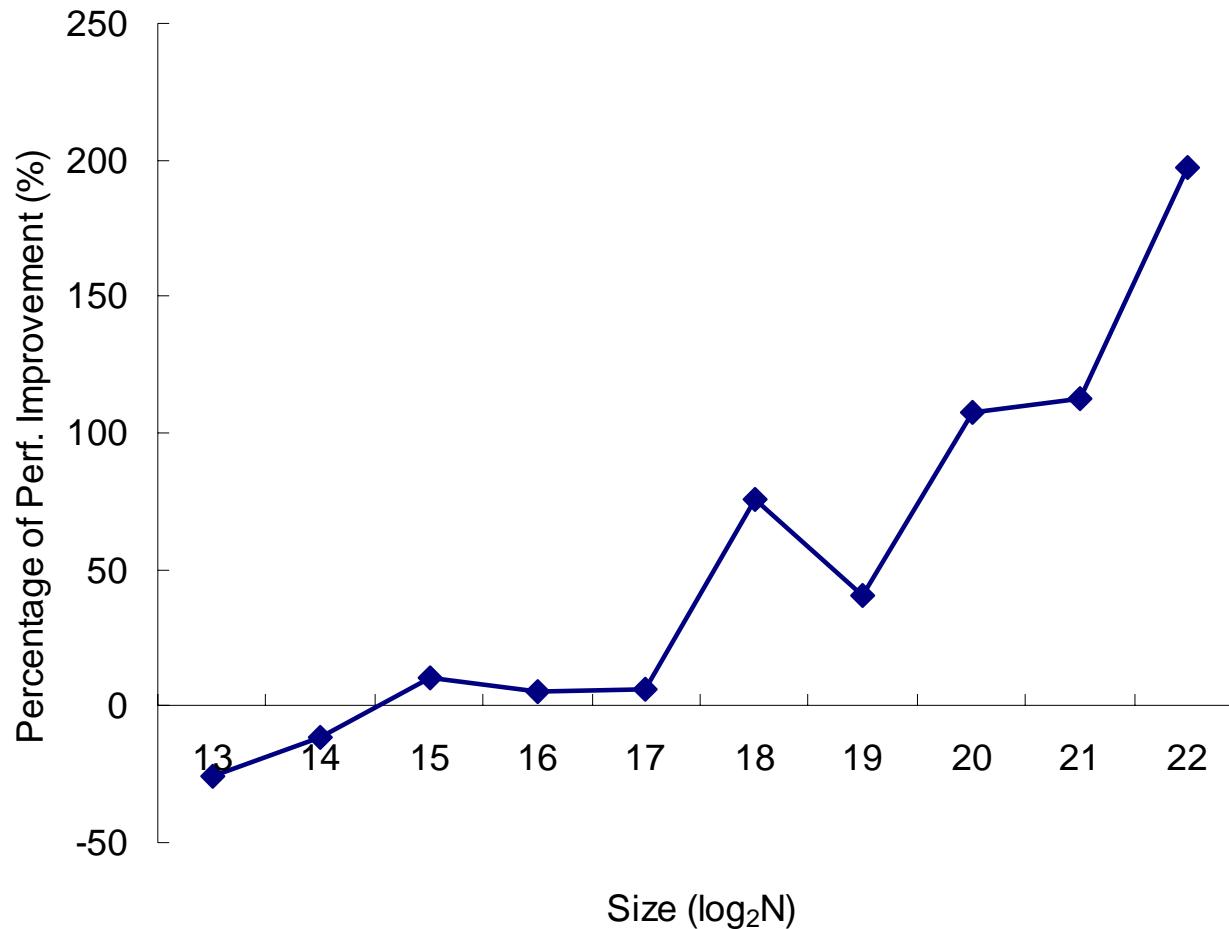
$$\frac{Mflops \text{ of } FFTDDL - Mflops \text{ of } FFTW}{\min(Mflops \text{ of } FFTDDL, Mflops \text{ of } FFTW)} \times 100 \text{ (%)}$$

- Experiments performed on 3 platforms
  - Alpha 21264 (500MHz, 64KB L1, 4MB L2, 1GB RAM)
  - MIPS R10000 (195MHz, 32KB L1, 4MB L2, 1GB RAM)
  - UltraSPARC3 (750MHz, 64KB L1, 4MB L2, 1GB RAM)

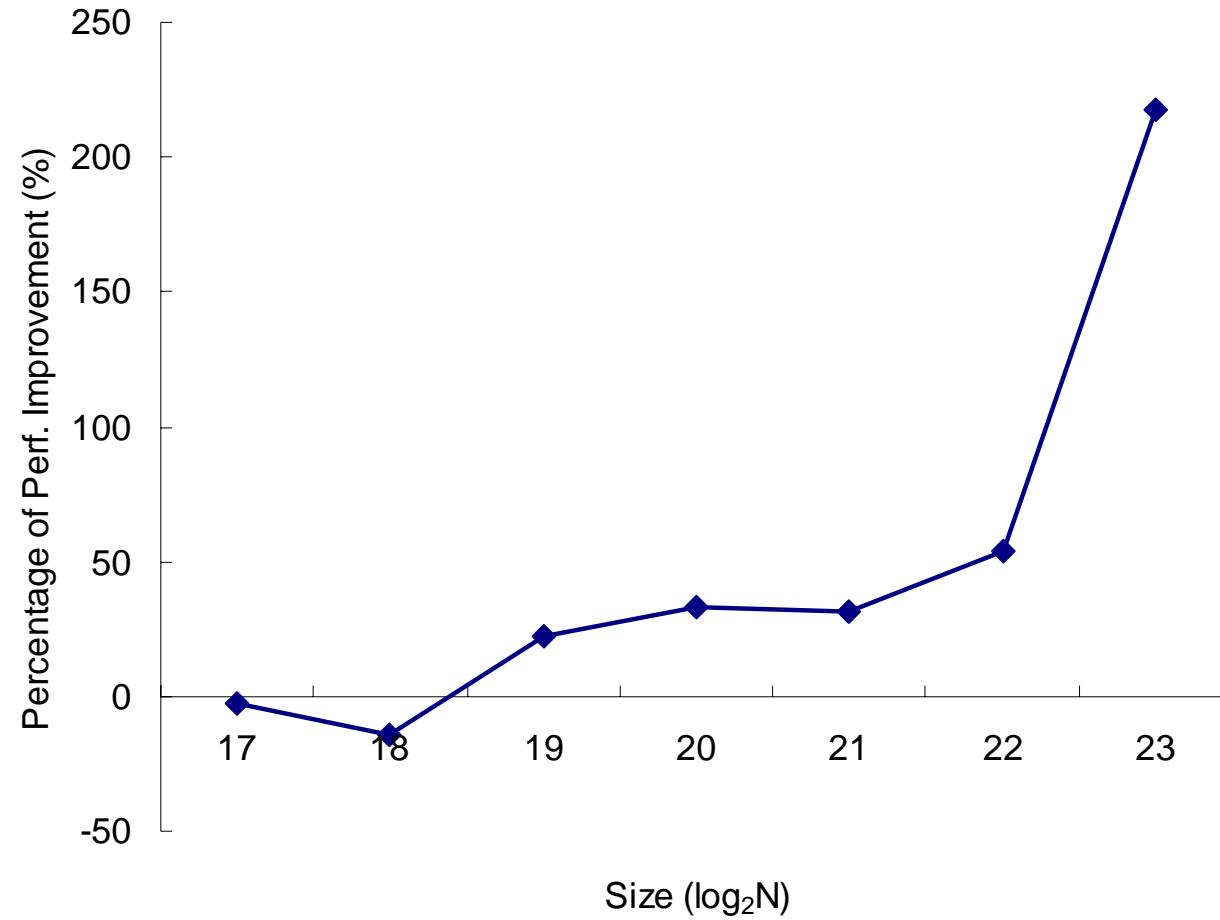
# Experimental Results: Alpha 21264



# Experimental Results: MIPS R10000



# Experimental Results: UltraSPARC 3



# Conclusion

---

- Array access using block layout
- TLB and cache performance improvements
- Tight bound on block size in ATLAS
- FFT performance optimization
- <http://advisor.usc.edu>

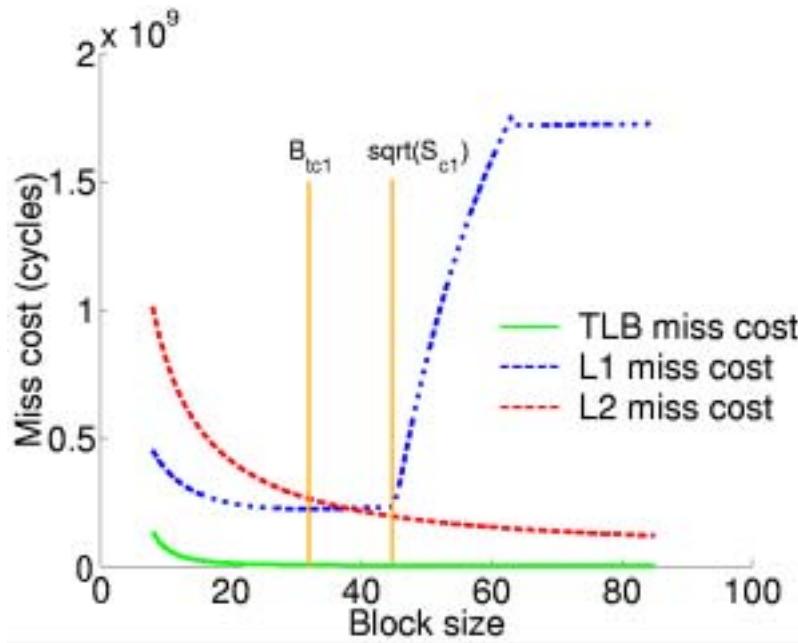
# Discussion



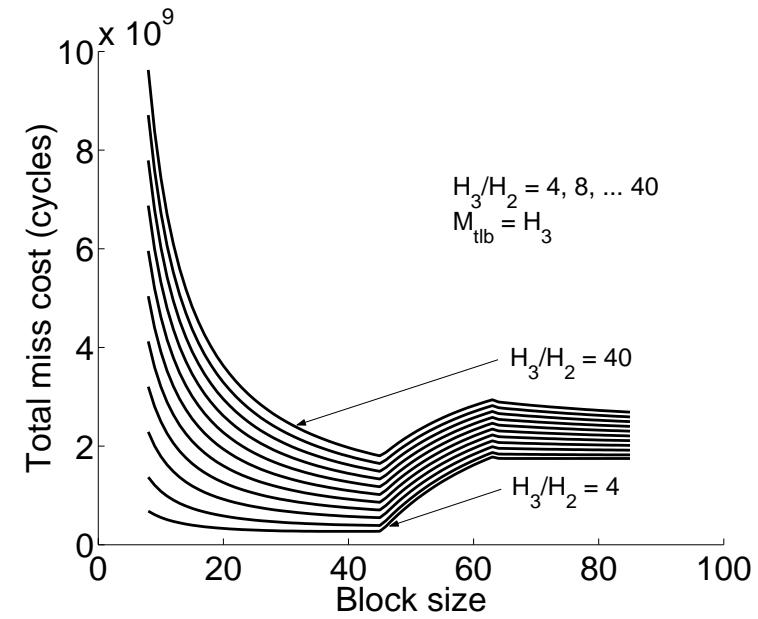
# Block Size Selection (2)

- The total miss cost for TMM

- (a) Miss cost of TLB, L1, and L2 cache



- (b) Total miss cost with various L2 miss penalty



# Optimal Block Size for TMM

- Simulation: using UltraSPARC II parameters
- Experiment: on UltraSPARC II

